Introducción

- Explicaremos un ejemplo de Goal Oriented Behaviour y de Goal Oriented Action Planner.

- Esta es una de muchas formas de implementarlas y esta es una de ellas.

- Vamos con el GOB, para empezar tendremos una lista de posibles Goals (metas) y Acciones con las que podemos satisfacerlas.

Goals

- Las Goals estarían compuestas por un nombre descriptivo que las identifique y un nivel de insistencia o peso.

- El nivel de insistencia sirve para dar prioridad a una goal u otra, la de más insistencia sería la primera que se intentaría resolver o aplacar.

- Este nivel se podría calcular a partir de información recopilada en nuestro blackboard.

- Ejemplo, podríamos tener las metas Saciado o Reconfortado, cuya insistencia podría calcularse en base a las propiedades de Hambre y Cansancio recopiladas en el blackboard.

- Otro ejemplo, una goal de Mantenerse a salvo, cuya insistencia se basa la propiedad de puntos de vida o el daño por segundo que estamos recibiendo.

- En una implementación orientada a objetos, probablemente Goal sea una clase base con las dos propiedades mencionadas, y cada Goal sería una especialización con un método que calcula su insistencia.

Acciones

- Las acciones, por lo general son la implementación de alguna habilidad, hechizo, ataque, etc. que puede ejecutar el personaje como Comer, Dormir, Curarse, Disparar, etc...

- En un sistema como el que estamos describiendo, también contendrá algún método o información que nos indica que goals puede satisfacer y en qué grado, es decir, cuánta insistencia restarán. Este valor nos servirá para hacer una estimación heurística, ya que en el que caso de que más de una acción resuelva una goal, escogeremos la que más nos satisfaga de una sola vez.

- El coste o rentabilidad de una acción es totalmente arbitrario, podemos agregar una propiedad específica o implementar un método que calcule cuánto nos cuesta ejecutar esa acción, y usar esa información para escoger la más rentable.
- Generalmente nuestro personaje o su blackboard tendrá una lista de acciones que puede realizar, al ser por personaje puede haber acciones específicas de su clase o rol que los demás no tengan.

- Por ejemplo en un juego de RPG, un sacerdote puede curar a otros jugadores o así, pero un pícaro no, en cambio el picaro tendrá la acción de camuflarse, que no estará disponible para otros jugadores. Si recordamos el ejemplo mencionado de goal "Mantenerse a salvo", estas acciones probablemente serían adecuadas para resolver esa meta.

- Es común tener este pool de acciones inicializado al empezar la partida según la información del personaje, pero pueden tener otros orígenes.

- Por ejemplo, según el nivel en el que estemos, dispondremos de habilidades únicas de ese escenario como activar ciertos mecanismos. Pueden ser incluso agregadas al vuelo al encontrarnos cerca de cierto objeto, incluso objetos que hayan hecho aparecer otros jugadores y que podamos usar, podrian agregar nuevas acciones mientras estemos cerca para resolver metas usando dichos objetos.

- También es muy útil agregar un método que nos devuelva si esa acción puede o no ser ejecutada en ese momento, ya sea por cooldown, o porque no tenemos algún componente necesario como maná, munición, etc.

Algoritmo básico

- Con esta información generalmente al empezar la partida tendremos:
  > Una lista de goals con un nivel de insistencia determinado.
  > Un pool de acciones de las que disponemos (que pueden o no ser ejecutadas).

- Repasamos la lista de goals en busca de la que tenga más insistencia.

- Una vez tenemos una referencia a esa meta, buscamos en nuestra lista de acciones una que resuelva nuestra meta.

- Al encontrar la primera acción que resuelva, guardamos tanto una referencia como su coste, y seguiremos buscando para compararla con la siguiente que encontremos, si la siguiente es más "barata" en términos heurísticos, esa será la escogida, hasta tener la que sea más rentable.

Conflictos

- Dado que las acciones pueden favorecer una meta pero también desfavorecer otras tras tras ejecutarlas, pueden producir efectos colaterales no deseados, dejados en una situación peor que la anterior.

- Para resolver esto, primero necesitamos que las acciones contenga información no sólo de las metas con las que reducen la insistencia, sino también con que metas la agravan.

- Teniendo estos datos podemos recurrir a ese método que nos dice cual es la acción más rentable, agregando a este cálculo un valor basado en la interacción de dicha acción con todas las otras metas en lugar de con la de mayor insistencia.

- Al comprobar una acción con todas las metas, el resultado es una lista de metas con los niveles de insistencia diferentes a la actual, resultantes de una supuesta ejecución de esa acción, una manera de saber si ese resultado global es mejor que otros es por ejemplo, sumar todos los niveles de insistencia (o como he visto en algunos ejemplos, los cuadrados de estas para evitar empates), teniendo de nuevo un valor con el que compararse con el resto de acciones, en este caso del de menor valor será el mejor (menor insistencia global).

- Con esta comparación, muy similar a cuando solo comparamos con una meta, podemos saber que acción es, en global, más eficaz ya que será la que menos perjudica a la insistencia de otras metas.

- Ejemplo, imagina las metas
  > Saciar Hambre con valor 4
  > y Evacuar (necesidad de ir al baño) con valor 3

-- La primera acción de la que disponemos es Comer Fruta, que nos da -2 de hambre y +1 de Evacuar en el baño:
  > nos deja con el estado Saciar hambre = 2
  > y Evacuar en el baño con valor = 4
  > Si sumamos los cuadrados de ambos resultados obtenemos 20 como valor de insistencia global.

-- La siguiente acción es Ir al baño, que únicamente nos da -3 de insistencia en Evacuar. Fíjate que si solo comprobáramos acciones que sacian nuestra meta con más insistencia como antes, está acción ni se tendría en cuenta, pero ahora estamos comparando con todas las metas, no solo con la de mayor valor.
  > al calcular nos deja con un valor de Hambre = 4, nuestra mayor meta no varía.
  > y un valor de Evacuar de 0,
  > Sumando los cuadrados obtenemos un valor de insistencia global de 16, mejor que del 20 de la anterior acción, y por lo tanto esta sería la acción elegida.

- Así que ante el panorama de tener bastante hambre, pero también algo de ganas de ir al baño, al evaluar qué acción escogemos primero iremos al baño a evacuar, y cuando volvamos a evaluar, nos pasaremos por la cocina a papear, algo que es bastante racional para un humano.

Translación implícita

- Hasta aqui hemos descrito el funcionamiento básico de un sistema de decisiones basado en métas, incluida la resolución de conflictos, pero si nos fijamos puede quedar la incógnita de como resolver el tema del movimiento.

- Todas las acciones son susceptibles de requerir que nos traslademos a un lugar concreto para realizarlas, este lugar puede ser dado por la meta o por la propia acción, y podemos definirlo como destino en nuestro blackboard por ejemplo.

- Típicamente este destino hará referencia a la posición de un enemigo, un mecanismo que debemos activar, un punto de encuentro o incluso nuestra posición actual en caso de acciones o metas que no requieran translación.

- Teniendo un destino fijado, cada acción puede contener un método que mida si estamos a la distancia adecuada para ejecutar la acción:
  > basándose en una propiedad de la acción que indique la distancia minima,
  > o en otras propiedades de nuestro personaje como el rango máximo de disparo de su arma a distancia o de melee, su área para interactuar con objetos, etc...

- Este método simplemente nos devolverá un booleano basándose en la distancia con nuestro actual destino y su criterio.

- Qué podemos hacer con esta información? facil, podemos crear una máquina de tres estados con los estados de Reposo, Movimiento y Ejecución.

- El estado de Reposo será en el que nuestra IA decida que meta y que acción ejecutará y a que destino debemos dirigirnos.

- Una vez tenemos una acción y un destino, saltaremos al estado de Movimiento, en este estado nuestro personaje se moverá hacia el destino especificado, y en cada tick comprobará con la acción si está a la distancia adecuada.

- En caso de estar en la distancia adecuada, saltaremos al estado de Ejecución donde finalmente ejecutaremos la acción.

- De esta forma cualquier acción puede tener implícita una traslación hacia donde tenemos que ejecutarla.

Tiempo como parte del coste heurístico

- Una mejora que podemos agregar a este sistema, que precisamente puede estar relacionado con la distancia es el tiempo, cuánto vamos a tardar en ejecutar una acción en concreto.

- Este tiempo puede formar parte del coste heurístico de la acción si en nuestro sistema necesitamos que las acciones cuesten el menor tiempo posible.

- Para tener en cuenta el tiempo en nuestro sistema, las metas no solo deben tener un nivel de insistencia, también deben tener una propiedad que indique cuánto sube la insistencia cada cierto tiempo.

- Por ejemplo, retornando al ejemplo de Saciar Hambre y Evacuar.
    > Saciar hambre tiene una insistencia de 4 y un incremento de insistencia de +1 cada 15 minutos.
    > Evacuar tiene insistencia 3 y sube +0.5 de insistencia cada 15 minutos.

- Ahora tenemos las siguientes acciones con los siguientes resultados

-- Comer Fruta, con -2 de Saciar Hambre y tarda 15 minutos
    > Como resultado nuestra hambre al cabo de 15 minutos será = 2 (al comer supuestamente anulamos el incremento de hambre)
    > Y necesidad de ir al baño subirá a 3.5.
    > Total de insistencia global = 16.25.

-- Otra acción, comer un plato de sopa, hambre -4 y tarda 1 hora
    > El resultado es que tenemos insistencia 0 en saciar hambre
    > e insistencia en evacuar de 5 debido a la hora transcurrida.
    > Total de insistencia global = 25.

-- Por último tenemos Ir al baño con -3 de insistencia de evacuar y 15 minutos de tiempo
    > Al no estar comiendo, la insistencia Saciar el hambre sube a 5
    > Insistencia de Evacuar baja a 0
    > Total de insistencia global = 25

- El resultado final es que primero comeremos la pieza de fruta y al reevaluar iremos al baño. En este sistema el resultado es diferente porque importaba el tiempo como parte de la eficiencia de la acción, al ser un ejemplo con necesidades humanas ciertamente parece ilógico. Sería más apropiado utilizarlo en una IA que no intente emular el razonamiento humano, sino más bien ser extremadamente eficiente en costes de tiempo.

Repaso

- Bién, hasta aquí hemos visto el funcionamiento básico de un sistema orientado a metas.
- Hemos visto como resolver conflictos para evitar efectos colaterales no deseados.
- Hemos agregado una máquina de estados simple para que el desplazarnos forme parte de nuestro sistema de forma implícita.
- Y finalmente hemos visto otra forma de calcular la eficiencia de nuestra acción cuando el tiempo es un factor importante.
- Para cerrar nuestro capítulo sobre comportamientos basados en metas, hablaremos del GOAP o Planificador de Acciones Orientado a Metas.

GOAP

- Básicamente el GOAP es el sistema que venimos explicando hasta ahora, solo que en lugar de obtener una sola acción al evaluar nuestras metas, obtenemos un plan, o en otras palabras, una cadena de acciones.

- Para entender de donde puede surgir esta necesidad vamos con un ejemplo, imaginemos un personaje de RPG, un Mago, con la siguiente situación.

> Según nuestro blackboard tenemos 5 unidades de mana para gastar.
> Una meta de Curarse con insistencia 4
> Una meta de Matar enemigo con insistencia 3
> y 3 acciones posibles,

-- Bola de fuego,

> con -2 de insistencia a matar enemigo y
> y un coste de 3 de maná

-- Cura menor

> con -2 de insistencia a Curarse y
> coste de maná = 2

-- Cura mayor

> con -4 de insistencia a curarse y
> coste de maná = 3

- La mejor combinación ahora mismo sería usar primero la cura menor y seguidamente la bola de fuego, pero si nos basamos en escoger primero la acción más rentable (menor insistencia global), la escogida sería la Cura mayor, dejándonos sin maná para ejecutar posteriormente la bola de fuego, y aunque nos hayamos curado, el combate podría acabar siendo desfavorable para nosotros, ya que nuestro enemigo no se ha visto debilitado en absoluto.

- En este ejemplo tan simple en el que exponemos poca información, podríamos coger el coste de maná y hacer que formará parte de su coste heurístico, escogiendo la opción con menos coste de maná que en este caso podría llevarnos a la situación correcta, pero si empezamos a agregar más factores externos al calculo heurístico este algoritmo podria resultar muy caótico, imagina, el maná que cuesta la acción si es que tiene coste, la situación actual de posibles rutas de escape, puertas abiertas, enemigos cercanos, enemigos ya alertados de nuestra presencia, etc.

- Aunque en la práctica un cambio de requisitos en nuestra IA pueda llevar a parchear nuestro algoritmo agregando este tipo de factores, en este ejemplo intentaremos mantenerlo lo mas limpio y agnóstico posible a estos factores externos.

- Dicho esto, la solución que se propone es que el sistema evalúe las metas encadenando más de una acción y no únicamente por separado.

- Hay que tener cuidado pues aunque el número de combinaciones sea finito, la cantidad de posibles combinaciones creceria de forma exponencial con cada acción disponible, con lo cual necesitamos un algoritmo de búsqueda de acciones eficiente.

- También hay que tener en cuenta que una vez escojemos una acción para la secuencia, otras acciones pueden quedar deshabilitadas según el estado en que nos dejan las anteriores acciones.

- En terminos de GOAP, lo que tendremos en nuestro blackboard es un "estado del mundo", formado por un conjunto de propiedades, las metas pasarían a definirse como estados del mundo a los que queremos llegar, y las acciones ahora contendrían dos listas de propiedades, unas serian los requisitos en el estado del mundo para poder ejecutarse, y otra los efectos que dicha propiedad causaría en el estado del mundo tras su ejecución.

- Aquí aparece una nueva entidad que estará presente en nuestro algoritmo de forma temporal y será un nodo, la cadena de acciones realmente será una cadena de nodo, y cada nodo contiene:

> La acción a ejecutar
> El estado temporal del mundo que resultaría tras ejecutar esa acción (lista de propiedades).
> El resultado de insistencia global acumulado.
> Una referencia al siguiente nodo.

- Para explicar el algoritmo y no ahogarnos en fórmulas complicadas de explicar sin una pizarra y unos dibujos, lo expresaré directamente con el ejemplo.

-- En nuestro estado actual del mundo tenemos:

> 5 unidades de mana
> Meta de Curarse con insistencia 4
> Meta de Matar enemigo con insistencia 3

-- Bola de fuego

> Prerrequisitos:
>> Mana >= 3
> Efectos:
>> -3 de Mana
>> -2 de Insistencia en Matar enemigo

-- Cura menor

> Prerrequisitos:
>> Mana >= 2
> Efectos:
>> -2 de Mana
>> -2 de Insistencia en Curarse

-- Cura mayor
  > Prerrequisitos:
    >> Mana >= 3
  > Efectos:
    >> -3 de Mana
    >> -4 de Insistencia en Curarse

- Lo que haremos será crear un árbol de posibles caminos por los que avanzar, posibles planes. Empezamos creando un nodo que representa el estado actual de nuestro mundo, sin acción asignada, y con el nivel de insistencia global actual. Con lo cual este primer nodo tendrá:
  > Ninguna acción asignada
  > Estado del mundo resultante:
    >> Mana = 5
    >> Curarse = 4
    >> Matar enemigo = 3
  > Nivel de insistencia global 25

- Ahora deberíamos calcular cual es el mejor nodo por el que ir, para rellenar la referencia al siguiente nodo, aquí tenemos las tres opciones referentes a las tres posibles acciones a ejecutar. Empezamos por la bola de fuego.

- Primer nodo a evaluar
  > Acción: bola de fuego
  > Estado del mundo resultante:
    >> Mana = 2
    >> Curarse = 4
    >> Matar enemigo = 1
  > Nivel de insistencia global 17

- Siguiente nodo a evaluar
  > Acción: cura menor
  > Estado del mundo resultante:
    >> Mana = 3
    >> Curarse = 2
    >> Matar enemigo = 3
  > Nivel de insistencia global 13

- Último nodo a evaluar
  > Acción: cura mayor
  > Estado del mundo resultante:
    >> Mana = 2
    >> Curarse = 0
    >> Matar enemigo = 3
  > Nivel de insistencia global 9

- Tras la primera evaluación el mejor nodo a ejecutar sería cura mayor, puesto que bola de fuego no se puede ejecutar, y cura menor no cambiaría el nivel de insistencia global en absoluto, este sería el primer nodo hoja candidato, pero como el nivel de insistencia global es mayor a 0, decidimos seguir buscando por los otros caminos.

- El siguiente mas rentable es el de cura menor, si lo encadenáramos con cura mayor volveriamos a obtener un nodo con insistencia global 9, como es mayor que 0, intentariamos otro camino, lo que daria pie a un nuevo nodo:

    > Acción: bola de fuego
    > Estado del mundo resultante:
        >> Mana = 0
        >> Curarse = 2
        >> Matar enemigo = 1
    > Nivel de insistencia global 5

- Como no tenemos más maná ya no podemos encadenar más nodos, y este sería el siguiente nodo hoja disponible, con un nivel de insistencia global 5.

- Como es mayor que 0 seguiríamos buscando y acabaríamos con otro camino igual pero con bola de fuego primero y cura menor después, cualquiera de los dos es candidato, pero si en nuestro algoritmo priorizamos intentar ir por el nodo más barato primero, el camino final será el compuesto por nodo inicial -> nodo cura menor -> nodo bola de fuego.

Documentación

http://www.amazon.es/Artificial-Intelligence-Games-Ian-Millington/dp/0123747317
http://alumni.media.mit.edu/~jorkin/gdc2006_orkin_jeff_fear.pdf (http://alumni.media.mit.edu/~jorkin/goap.html)